

PRACTICAL: 5

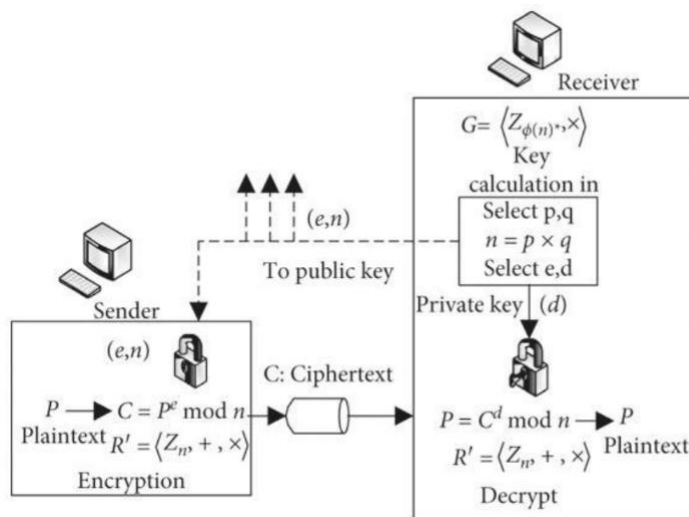
AIM:

XYZ Bank implements the encryption technique shown in figure to secure transactions between its servers and customers. The bank uses the following setup:

1. The server's public key is $n=119$ and $e=5$.
2. Customers encrypt sensitive information, such as their PINs, using this public key before sending it to the server.

A customer wants to send their 2-digit PIN, $M=31$, to the bank.

However, an attacker intercepts the encrypted message, which is $C=92$ and algorithm used. The attacker is determined to decrypt the ciphertext and discover the PIN.



Implement the above scenario.

THEORY:

Key Generation

Choose two prime numbers p and q .

Compute $n = p \times q$

Calculate $\phi(n) = (p - 1) \times (q - 1)$.

Encryption

The plaintext M is encrypted using the public key:

CODE:

```
import java.util.*;
public class crns{

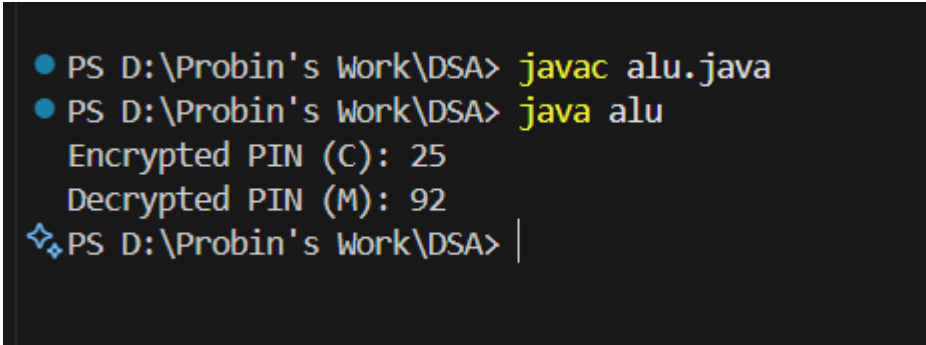
    public static long modExp(long base, long exp, long mod) {
        long result = 1;
        while (exp > 0) {
            if (exp % 2 == 1) {
                result = (result * base) % mod;
            }
            base = (base * base) % mod;
            exp /= 2;
        }
        return result;
    }

    public static long modInverse(long e, long phi) {
        for (long d = 1; d < phi; d++) {
            if ((e * d) % phi == 1) {
                return d;
            }
        }
        return -1;
    }

    public static void main(String[] args) {
        long p = 9, q = 18;
        long n = p * q;
        long e = 5;
        long phi = (p - 1) * (q - 1);
        long d = modInverse(e, phi);
        if (d == -1) {
            System.err.println("Error computing modular inverse. Exiting.");
            return;
        }

        long M = 31;
        long C = modExp(M, e, n);
        System.out.println("Encrypted PIN (C): " + C);

        long intercepted_C = 92;
        long decrypted_M = modExp(intercepted_C, d, n);
        System.out.println("Decrypted PIN (M): " + decrypted_M);
    }
}
```

OUTPUT:

```
PS D:\Probin's Work\DSA> javac alu.java
PS D:\Probin's Work\DSA> java alu
Encrypted PIN (C): 25
Decrypted PIN (M): 92
❖ PS D:\Probin's Work\DSA> |
```

LATEST APPLICATIONS:

1. Secure Web Browsing (HTTPS): RSA is widely used in SSL/TLS protocols to ensure secure communication over the internet. It's used for key exchange during the handshake process in HTTPS, allowing web servers and browsers to securely share encryption keys.
2. Digital Signatures: RSA is commonly used in digital signatures to authenticate the identity of the sender and verify the integrity of messages. It's used in email services, software distribution, and legal documents to ensure that a message or file hasn't been altered.
3. VPNs and Secure Communication Channels: RSA is used in Virtual Private Networks (VPNs) to encrypt the data transferred between users and networks, ensuring that sensitive data remains secure even over insecure networks like the internet.
4. Cloud Security: In cloud computing environments, RSA is utilized for securing data transfer between clients and cloud servers. It's also used in data storage encryption to prevent unauthorized access to sensitive data stored on cloud platforms.

LEARNING OUTCOME:

- Learn RSA Algorithm – Explore key generation, encryption, and decryption using modular arithmetic.
- Develop Cryptographic Functions – Implement modular exponentiation and modular inverse in Java.
- Expand Cybersecurity Expertise – Understand data protection strategies in practical applications.

REFERENCES:

1. <https://www.geeksforgeeks.org/rsa-algorithm-cryptography/>
2. <https://www.youtube.com/watch?v=VF3AHG0T9ec>